consider adapting the SIMD processor taught in Dieffenderfer for use in combination with the switch in Steely (assuming strictly arguendo that the switch of Steely could be considered to be a state engine) to arrive at Applicant's claimed combinations.

Accordingly, reconsideration and withdrawal of the rejection of claims 19-38 under 35 U.S.C. § 103(a) over Steely in view of Dieffenderfer are respectfully requested.

Other differences exist between Applicant's claimed state engine and the components described in Steely, e.g., the switch and the arbiter. New claims 39-46 have been added to provide additional claim coverage which further describe these differences. A general description of these differences is provided below to assist in the understanding of differences between Applicant's claims and Steely.

According to exemplary embodiments, a state engine provides multiple local shared memories whereas Steely only describes one shared memory. These multiple shared memories allow for a bandwidth that is the sum of the bandwidths of all the memories over the state engine. This additionally allows for the partitioning of processing functions, as well as having a plurality of state elements (see new claims 42 and 45). Segregation of function can also be performed whereby the state engine separates shared state from shared memory access which can lead to efficient scalability. In support of scalability, among other reasons according to exemplary embodiments, one or more state engines can be applied to a system bus and operate

allowable for reasons of their own. For example, Applicant's claim 31 combination states the following:

"A state engine as claimed in claim 19, wherein each said state element means comprises a local memory for said shared state, an arithmetic unit adapted to perform the operation on said state in said local memory, and command and control logic to control said operation".

Regarding Applicant's claim 31 combination, it is respectfully submitted that the arbiter 240 of Steely that is alleged to correlate to Applicant's "state element" does not describe having "an arithmetic unit adapted to perform the operation on said state in said local memory". By way of contrast, the arbiter 240 forwards commands, with the processor that sent the command having to wait for any data to return prior to the processor (and not the arbiter) having the option of modifying the data and writing it back. Accordingly it does not appear that the arbiter 240 needs or has the function of "an arithmetic unit adapted to perform the operation on said state in said local memory" which can only be found, among other features, in Applicant's claim 31 combination.

As described above, Steely does not disclose all of the features in Applicant's claimed combinations. The Official Action also uses the Dieffenderfer patent and, more specifically, the Official Action uses Dieffenderfer because Steely does not specifically teach the use of receiving requests from a parallel processor. Dieffenderfer does describe a parallel array processor (see Figure 4 and col. 12, lines 51-53) which may be configured as a SIMD subsystem. However, Dieffenderfer fails to teach or suggest that the SIMD processor modifies shared state in a shared memory for other processors. Therefore, it is respectfully submitted that one of ordinary skill in the art would not

This is conceptually quite different from Applicant's system where the processors send commands to the state engine directing it how to update the maintained state. Additionally, in Applicant's state engine there is no arbitration between the processors' requests, instead the processors' requests are queued in the order in which they arrive. Therefore, it follows that the position taken in the Official Action that the arbiter 240 constitutes Applicant's state element is a non-sequitur.

For another example with respect to Applicant's claim 19 combination, consider the claim 1 element of "said at least one state element adapted to operate, atomically, on said shared state". With respect to this claim 1 element, it is respectfully submitted that Steely's only apparent reference to "atomic" operation is in col. 9, lines 1-7, where Steely describes that atomic operation relates to the serialization function of the coherence controller 180 and the Arb bus 170. Neither the coherence controller 180 nor the Arb bus 170 are components of the switch 200 which the Official Action equates to Applicant's claimed state engine. Therefore, Steely does not disclose Applicant's claim 1 combination of "said at least one state element adapted to operate, atomically, on said shared state" at least because the components which perform anything "atomically" (as used herein) are not in the switch 200 which is alleged to correspond to Applicant's claimed state engine. Similar comments apply to independent claims 32, 35, 36, and 38.

The dependent claims are allowable at least for the reasons described above for the independent claims from which they depend. Additionally the dependent claims are

of Steely is the same as Applicant's claimed state engine which makes this argument relevant. That is in each independent claim the least one state element and the memory form a part of the state engine. By way of contrast, assuming (strictly arguendo) that the switch in Steely could be considered a state engine, the memory of Steely is separate from the switch and as such is not a part of the switch and does not meet the claims.

Regarding the independent claims, more arguments will now be described to further explain the differences between Steely, Dieffenderfer and Applicant's claims. Additionally, to further clarify these differences, new dependent claims 39-46 have been added (which will be described in more detail below). Initially, it is respectfully submitted that the Official Action incorrectly equates the switch 200 in Steely with Applicant's claimed state engine. From Figures 1 and 2 of Steely and the associated text, it is respectfully submitted that the switch 200 cannot be regarded as a state engine at least because no operations are performed on the data passing through the various queues connected to the ports 202-206 nor in the arbiter 240, i.e., the data does not change. For example, Steely describes the function of the switch 200 in columns 5-7 as being to queue the requests, the probes and the requested data, while the function of the arbiter is to arbitrate among the input queues to grant access to the Arb bus 170 according to, for example, a round-robin protocol. In other words, Steely simply queues and arbitrates updates to memory from the processors.

Action at page 3, second full paragraph, but also a request that includes at least a command directing at least one state element means on how to perform an operation on a shared state, and a memory connected to the at least one state element means and configured to store the shared state.

Dieffenderfer has been considered but does not cure the deficiencies of Steely discussed above with regard to independent Claim 19.

In response to the above arguments, the Official Action dated October 30, 2008 presents two counter arguments which, it is respectfully submitted, are not sufficient to establish a prima facie case of obviousness for at least the following reasons. Firstly, the Official Action references Steely (Fig. 2; col. 9, lines 26-51; and col. 10, lines 24-54) in an attempt to show that Steely teaches, for example, Applicant's claim 1 combination. However, as Steely does not teach a programmable entity capable of executing shared memory instructions, it follows that the command issued in Steely for routing the data cannot reasonably be considered an "operation" on a shared state as claimed by Applicant (further discussion of differences related to the claimed "operation" is presented below).

Secondly, the Official Action takes the position that "it is noted that the features upon which applicant relies (i.e., a memory that stores the shared state being included in the switch 200 and the memory 150 that stores the shared state is outside the switch 200) are not recited in the rejected claim(s)". While it is true that the claimed combinations do not recite a switch, it is the Official Action's position that the switch 200

does not teach or suggest processors sending commands to the state engine directing it regarding how to update the memory.

In this regard, if a processor in Steely needs to increment a shared memory location, then the processor needs to read the data from the memory, increment the value, write it back to the memory, and then issue a memory barrier instruction to synchronize this update with all the other processes. However, this operation delays the other processors from accessing that memory location until the first processor has finished the complete sequence read-modify-write operation, as illustrated in Figure 5(b).

To the contrary, the claimed invention avoids this latency because any number of state processors may instruct the state engine via the requests, for example, to increment the same shared state and the processors may immediately continue with other processing. Thus, the processors do not have to wait for the whole operation read-modify-write to be performed, as illustrated by Figure 5(a).

In addition, the independent claims recite that the state engine also includes a memory connected to the at least one state element means and configured to store the shared state. Steely is silent about a memory that stores the shared state being included in the switch 200. On the contrary, Figure 2 of Steely clearly shows that the memory 150 that stores the shared state is outside the switch 200.

Therefore, the undersigned respectfully submits that Steely is lacking not only the request made by the parallel processor, as acknowledged by the outstanding Office

Turning to the applied art, Steely discloses a technique that reduces a latency of a memory barrier operation used to impose an inter-reference order between sets of memory reference operations issued by a processor to a multiprocessor system having a shared memory. More specifically, Figure 2 shows a local switch 200 communicating with a plurality of processors P1 to P4 such that inputs from the processors are serialized before being sent to a shared memory 150. Figure 2 shows that the switch 200 includes an arbiter 240 that arbitrates, among input queues from the processors P1 to P4, to grant access to the Arb bus 170, where the requests are ordered into a memory reference request stream. The arbiter 240 selects the request stored in the input queues for access to the bus in accordance with an arbitration policy, such as a conventional round-robin algorithm, as disclosed in Steely in the paragraph bridging columns 6 and 7.

Thus, the device of Steely does not teach or suggest that a request from any of the processors P1 to P4 includes at least a command directing the arbiter 240 regarding how to perform an operation on the shared memory 150, as recited, among other things, by the independent Claims.

In other words, the arbitration system of Steely does not perform an operation on the shared memory 150 based on a command from the processors P1 to P4 but rather acts based on the round-robin algorithm, which is not provided by the processors P1 to P4. Steely simply queues and arbitrates updates to memory from the processors and

REMARKS

Favorable reconsideration of this application as presently amended and in light of the following discussion is respectfully requested.

Claims 19-46 are pending in the present application. New dependent claims 39-46 have been added.

In the outstanding Office Action, Claims 19-38 were rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over Steely, Jr. et al. (U.S. Patent No. 6,088,771, herein "Steely") in view of Dieffenderfer et al. (U.S. Patent No. 5,822,608, herein "Dieffenderfer"), which ground of rejection is respectfully traversed at least for the following reasons.

Briefly recapitulating, independent Claim 19 is directed to a state engine that receives multiple requests from a parallel processor for a shared state. The state engine includes at least one state element means, the at least one state element means adapted to operate, atomically, on the shared state in response to a request made by the parallel processor. The request includes at least a command directing the at least one state element means on how to perform an operation on the shared state. The state engine also includes a memory connected to the at least one state element means and configured to store the shared state.

The claimed one state element unit advantageously achieves, for example, faster access for the parallel processor to the shared state, as shown for example in Figure 5(b) of the present application and its corresponding description in the specification.